



暨南大學
JINAN UNIVERSITY

Intelligent Flight System

(First semester for Year 2024-2025)

Course Name: Database System

Course Type: Optional

Student Name: 段瑞豪

Student ID: 2022180163

College: International School

Department: _____

Major: Computer Science and Technology

Professor: 吴汉瑞

December 11th, 2024

Contents

1. Abstract.....	3
2. Introduction	3
3. Objectives	3
4. System Design	4
4.1 Entity-Relationship (ER) Model.....	4
4.2 Database Schema	5
5. Implementation.....	7
5.1 Technologies Used	7
5.2 File Structure.....	7
5.3 System Functions	7
6. Testing and Result.....	8
7. Conclusion.....	10
8. Future Enhancements	10
9. References.....	11

1. Abstract

The Flight Simulation Database System is designed to manage user data and flight logs efficiently. This project focuses on implementing a relational database using MySQL to store and retrieve user information, such as experience level, profile settings, and flight activity logs. The system is developed in Java, adhering to best database and programming practices, while ensuring data integrity and usability.

2. Introduction

Database systems play a critical role in managing large amounts of data in modern applications. The **Flight Simulation Database System** provides a structured solution for managing users and their flight records within a simulation program.

The system allows for core operations:

- **Create:** Add users and flight logs
- **Read:** Retrieve user and flight log data
- **Update:** Modify user experience levels
- **Delete:** Remove users or flight records

This report documents the design, implementation, and testing of the database system, with a focus on data management, integrity constraints, and program usability.

3. Objectives

The objectives of this project are as follows:

1. To design and implement a relational database for flight simulation.
2. To develop a Java-based application that interacts with the database.
3. To enforce **data integrity** using primary and foreign key constraints.
4. To provide a simple interface for performing CRUD operations.
5. To test and validate the system for accuracy and efficiency.

4. System Design

4.1 Entity-Relationship (ER) Model

The system is modeled using two primary entities:

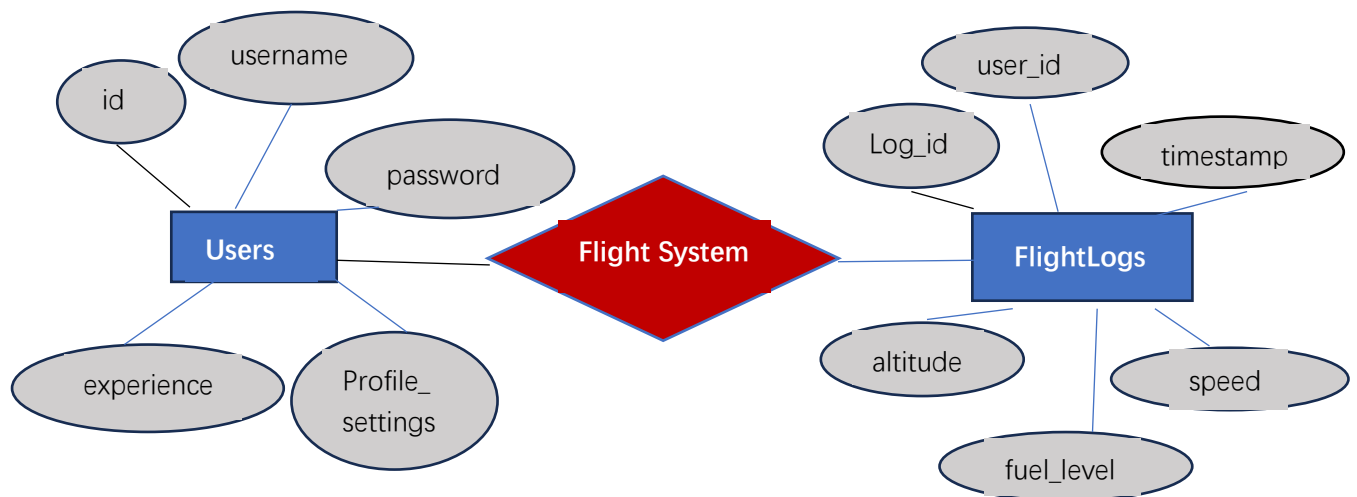
1. **Users**
2. **FlightLogs**

ER Diagram

The relationship between entities is shown below:

Entities and Relationships:

- **Users Table:**
 - Attributes: id (PK), username, password, experience, profile_settings
- **FlightLogs Table:**
 - Attributes: log_id (PK), user_id (FK), timestamp, altitude, speed, fuel_level
- **Relationships:**
 - FlightLogs.user_id references Users.id (one-to-many relationship).



The **Entity-Relationship (ER) Model** of the Flight Simulation Database System is designed to capture the relationships between core entities while enforcing data integrity through appropriate constraints. The model comprises two main entities: **Users** and **FlightLogs**. These entities interact with each other through a one-to-many relationship, ensuring that each user can have multiple flight logs while maintaining referential consistency.

The **Users** table represents individuals participating in the flight simulation system. It contains

essential information about each user, including a unique identifier (id), a username for identification, a password for securing access, and attributes to record their experience level and profile settings. The primary key for the Users table is the id column, which ensures that each user in the system has a distinct, non-null identifier. This uniqueness allows the table to serve as the parent entity in the relationship with the FlightLogs table.

The **FlightLogs** table serves as a record of flight activities associated with the users. It tracks information such as flight altitude, speed, and fuel levels, along with a timestamp to indicate when the log was recorded. Each entry in the FlightLogs table is uniquely identified by the log_id column, which serves as its primary key. To link flight logs to users, the user_id column in the FlightLogs table is defined as a **foreign key** that references the id column in the Users table. This foreign key constraint establishes a one-to-many relationship: a single user in the Users table can be associated with multiple flight logs in the FlightLogs table, but each flight log can belong to only one user.

This relationship between the two tables ensures data integrity and consistency. For instance, the foreign key constraint prevents the insertion of a flight log with a user_id that does not exist in the Users table. It also ensures that if a user is deleted from the database, the related flight logs can be handled appropriately to avoid orphaned records.

The ER model thus provides a robust and structured foundation for managing user data and associated flight logs. By separating user attributes and flight activity data into two related tables, the design achieves both **normalization** and **efficiency**, avoiding redundancy while enabling seamless querying and management of the data.

4.2 Database Schema

The following tables were created in the **MySQL** database:

Users Table

Column	Type	Constraints
id	INT	PRIMARY KEY, AUTO_INCREMENT
username	VARCHAR(50)	NOT NULL, UNIQUE
password	VARCHAR(50)	NOT NULL
experience	VARCHAR(50)	
profile_settings	VARCHAR(50)	

FlightLogs Table

Column	Type	Constraints
log_id	INT	PRIMARY KEY, AUTO_INCREMENT
user_id	INT	FOREIGN KEY REFERENCES Users(id)

Column	Type	Constraints
timestamp	DATETIME	DEFAULT CURRENT_TIMESTAMP
altitude	INT	
speed	INT	
fuel_level	INT	

Constraints

The **Users** table has the following constraints:

- A **Primary Key** constraint on the id column to uniquely identify each user, with auto-increment enabled.
- A **Unique Constraint** on the username column to ensure no duplicate usernames exist.
- A **Password Length Constraint** using CHECK, which ensures the password column has a minimum length of 8 characters.
- A **NOT NULL Constraint** on critical columns (username, password, experience, and profile_settings) to ensure that these fields cannot contain null values.

The **FlightLogs** table has the following constraints:

- A **Primary Key** constraint on the log_id column to uniquely identify each flight log, with auto-increment enabled.
- A **Foreign Key Constraint** on the user_id column, which references the id column in the Users table. This ensures referential integrity between the two tables.
- **ON DELETE CASCADE** behavior for the user_id foreign key, so when a user is deleted, all their corresponding flight logs are automatically deleted.
- **ON UPDATE CASCADE** behavior for the user_id foreign key, so changes to the id in the Users table are reflected in FlightLogs.
- An **Altitude Range Constraint** using CHECK, which ensures the altitude column contains values between 0 and 60,000 feet.
- A **Speed Constraint** using CHECK, which restricts the speed column to values between 0 and 1,000 knots.
- A **Fuel Level Constraint** using CHECK, which ensures the fuel_level column has values between 0 and 100,000 units.
- A **Default Timestamp Constraint** on the timestamp column, which automatically assigns the current timestamp during insertion.
- A **NOT NULL Constraint** on the altitude, speed, fuel_level, and user_id columns to ensure these fields are always populated.

The **relationship constraints** include:

- A **Foreign Key Integrity Constraint** that ensures no orphan records exist in the FlightLogs table by enforcing that user_id must match an existing id in the Users table.
- A **One-to-Many Relationship Validation** between the Users and FlightLogs tables, where a single user can have multiple flight logs, but each flight log corresponds to only one user.

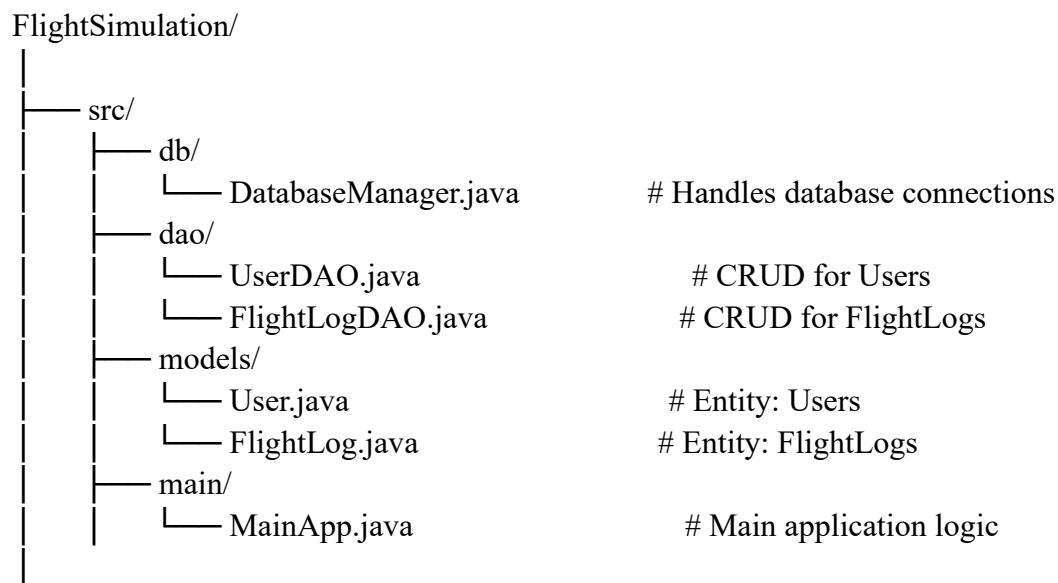
5. Implementation

5.1 Technologies Used

- **Programming Language:** Java
- **Database Management System:** MySQL
- **IDE:** IntelliJ IDEA
- **Connector:** MySQL Connector/J
- **File System:** Text-based configuration

5.2 File Structure

The project follows a structured modular design:



5.3 System Functions

1. Add User

Allows new users to be inserted into the database.

2. View Users

Displays all users in the database.

3. Update Experience Level

Updates the experience field for a specific user.

4. Delete User

Deletes a user based on the user ID.

6. Testing and Result

The system was tested with various test cases: The results can be observed in the provided snapshots

Test Case 1: Adding a User

- **Input:** username = DuanHao, experience = UpperLevel, profile_settings = "theme": "gold"
- **Result:** User successfully added.

Test Case 2: Viewing All Users

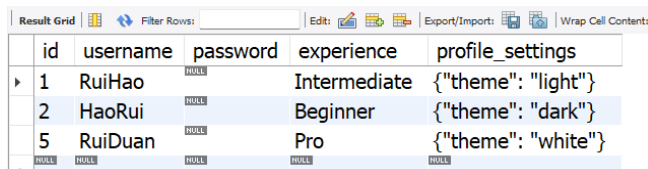
- **Expected:** All user records displayed.
- **Actual:** Records displayed correctly.

Test Case 3: Updating Experience

- **Input:** Update user ID 2 experience to Advanced.
- **Result:** Experience updated successfully.

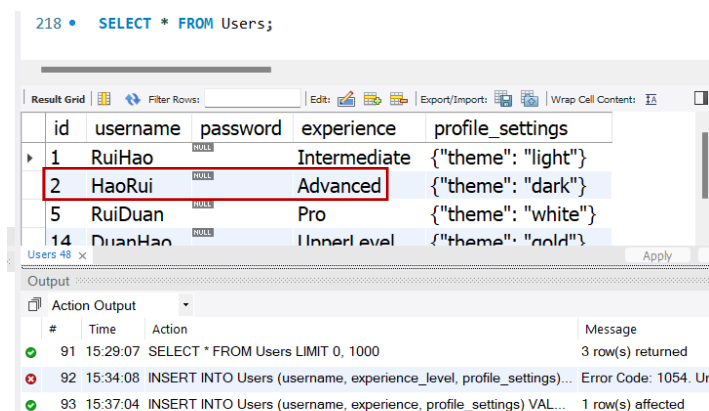
Test Case 4: Delete a user

- Attempted to delete a user_id.
- **Result:** The deleted user disappeared from the user table.



id	username	password	experience	profile_settings
1	RuiHao	H00L	Intermediate	{"theme": "light"}
2	HaoRui	H00L	Beginner	{"theme": "dark"}
5	RuiDuan	H00L	Pro	{"theme": "white"}

Figure: Original user table in the MySQL workbench



id	username	password	experience	profile_settings
1	RuiHao	H00L	Intermediate	{"theme": "light"}
2	HaoRui	H00L	Advanced	{"theme": "dark"}
5	RuiDuan	H00L	Pro	{"theme": "white"}
14	DuanHao	H00L	Inner level	{"theme": "gold"}

Output

#	Time	Action	Message
91	15:29:07	SELECT * FROM Users LIMIT 0, 1000	3 row(s) returned
92	15:34:08	INSERT INTO Users (username, experience_level, profile_settings)...	Error Code: 1054. Ur
93	15:37:04	INSERT INTO Users (username, experience, profile_settings) VAL...	1 row(s) affected

Figure: Successfully updated user ID 2 experience to Advanced

```
--- Flight Simulation Database ---
1. Add User
2. View All Users
3. Update User Experience Level
4. Delete User
5. Exit
Choose an option: 2

--- All Users ---
Re-established database connection.
ID: 1
Username: RuiHao
Experience: Intermediate
Profile Settings: {"theme": "light"}
-----
ID: 2
Username: HaoRui
Experience: Beginner
Profile Settings: {"theme": "dark"}
-----
ID: 5
Username: RuiDuan
Experience: Pro
Profile Settings: {"theme": "white"}
-----

--- Flight Simulation Database ---
1. Add User
2. View All Users
3. Update User Experience Level
4. Delete User
5. Exit
Choose an option:
```

Figure: Original user table viewed in IntelliJ

```
--- Flight Simulation Database ---
1. Add User
2. View All Users
3. Update User Experience Level
4. Delete User
5. Exit
Choose an option: 3
Enter user ID: 2
Enter new experience level: Advanced
User experience updated successfully.
```




Figure: Adding new user information

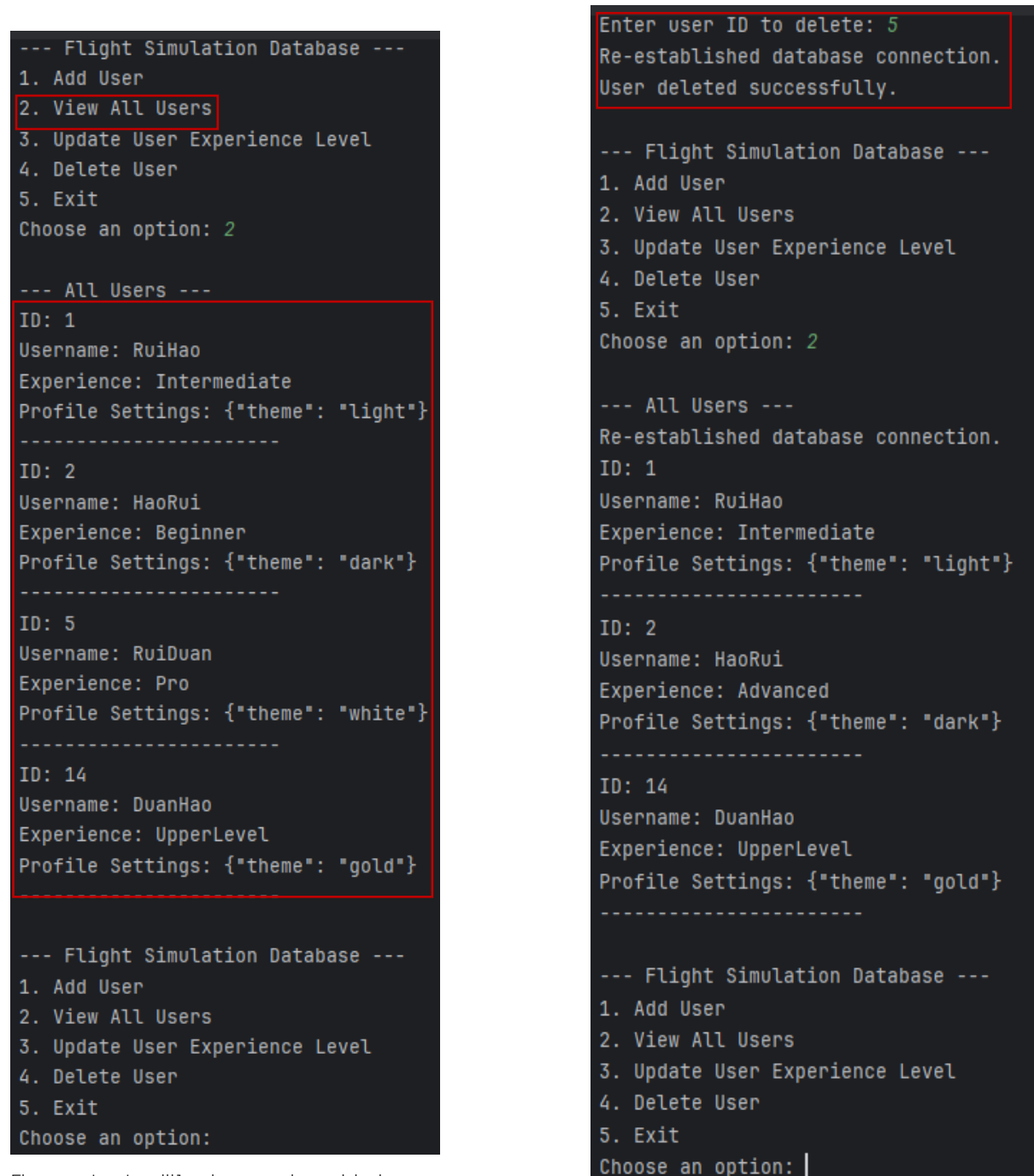


Figure: In IntelliJ, the newly added user information can be observed

Figure: User ID 5 successfully deleted from the

7. Conclusion

The **Flight Simulation Database System** successfully manages user information and flight logs, demonstrating robust database design and seamless integration with a Java-based application. The use of foreign key constraints ensures referential integrity, and the modular code design enhances scalability.

8. Future Enhancements

1. Develop a graphical user interface (GUI).
2. Add user authentication for improved security.
3. Include analytics and reporting for flight logs.

9. References

- Elmasri, R., & Navathe, S. B. (2016). *Fundamentals of Database Systems* (7th ed.). Pearson Education.
- Silberschatz, A., Korth, H. F., & Sudarshan, S. (2020). *Database System Concepts* (7th ed.). McGraw-Hill Education.
- Connolly, T., & Begg, C. (2014). *Database Systems: A Practical Approach to Design, Implementation, and Management* (6th ed.). Pearson.